

IN THE CLAIMS:

Please amend the claims as follows:

1. (Currently Amended) A computer-implemented method for debugging code, comprising:

receiving a selection of a target call site, the target call site comprising a call to a routine; and

setting at least one run into breakpoint configured to halt subsequent execution only when an execution path arrives at the routine from the target call site;

upon determining that the execution path has arrived at the routine, halting execution of the code; and

performing a step into operation to step into the routine.

2. (Original) The method of claim 1, wherein:

the selection is received while execution is halted at a first point in the code;

the target call site is located at a second point in code; and

the first point is a first statement in the code and the second point is a second statement in the code.

3. (Original) The method of claim 1, wherein the code is object-oriented and the target call site is a method of the object-oriented code.

4. (Original) The method of claim 1, wherein setting the at least one run into breakpoint comprises setting the run into breakpoint on an instruction calling the routine.

5. (Original) The method of claim 1, wherein setting the at least one run into breakpoint comprises setting the run into breakpoint at each entry point to the routine.

6. (Original) The method of claim 5, further comprising:

encountering the at least one run into breakpoint;

determining whether the routine is entered from the selected target call site; and if so, halting execution of the code.

7. (Currently Amended) A computer-implemented method for debugging code, comprising:

receiving a selection of a target call site in the code, the target call site comprising a call to a routine;

establishing a breakpoint at an entry point to the selected target call site;

determining call context information identifying a location of the selected target call site in the code;

upon encountering the breakpoint during execution of the code, determining whether the routine is called from the selected target call site based on the call context information; and

if so, halting execution of the code and then automatically performing a step into operation to step into the routine.

8. (Original) The method of claim 7, wherein the target call site is located at a first point in the code and the selection of the target call site is received while execution is halted at a second point in the code.

9. (Original) The method of claim 7, wherein the call context information unambiguously identifies the code.

10. (Original) The method of claim 7, wherein the code is object-oriented and the target call site is a method of the object-oriented code.

11. (Currently Amended) The method of claim 7, wherein determining whether the routine is called from the selected target call site based on the ~~stored~~ call context information comprises comparing the ~~stored~~ call context information to the selected content of a call stack.

12. (Original) The method of claim 11, wherein the routine is determined to have been called from the selected target call site if the stored call context information matches the selected content of the call stack.

13. (Original) The method of claim 11, wherein the selected content of the call stack is a call to the routine and wherein the routine is determined to have been called from the selected target call site if the stored call context information matches the selected content of the call stack.

14. (Currently Amended) A computer-implemented method for debugging code, comprising:

(a) receiving a selection of a target call site in the code, the target call site comprising a call to a routine having a plurality of entry points;

(b) establishing a breakpoint at each of the plurality of entry points;

(c) determining call context information uniquely identifying the selected target call site; and

(d) for each of the breakpoints encountered during execution of the code:

determining whether the routine is called from the selected target call site based on the call context information; and

if so, halting execution of the code and then performing a step into operation to step into the routine.

15. (Currently Amended) The method of claim 14, further comprising:

repeating each of the steps (a)-(d) for a plurality of target call sites each having a routine with a plurality of entry points, so that each target call site has a plurality of associated breakpoints set at each of the associated plurality of entry points; and

automatically removing ~~only the~~ at least one of the plurality of associated breakpoint associated with a particular selected target call site upon determining that the routine is called from the particular selected target call site based on the call context information.

16. (Currently Amended) The method of claim 14, wherein determining whether the routine is called from the selected target call site based on ~~the stored call context information comprises~~ comparing the ~~stored~~ call context information to selected content of a call stack.

17. (Original) The method of claim 16, wherein the breakpoint is determined to be called from the selected target call site if the stored call context information matches the selected content of the call stack.

18. (Original) The method of claim 16, wherein the selected content of the call stack is a call to the routine and wherein the routine is determined to be called from the selected target call site if the stored call context information matches the selected content of the call stack.

19. (Cancelled)

20. (Currently Amended) The method of claim 14 ~~49~~, further comprising, prior to identifying, determining that the method involves an object.

21. (Currently Amended) The method of claim 14 ~~49~~, wherein identifying comprises traversing a class hierarchy.

22. (Currently Amended) The method of claim 14 ~~49~~, wherein identifying comprises traversing a class hierarchy and locating each matching member method according to the selected target call site.

23. (Cancelled)

24. (Cancelled)

25. (Cancelled)

26. (Cancelled)

27. (Currently Amended) A computer readable storage medium containing a program which, when executed, performs an operation for debugging code, comprising:

receiving a selection of a target call site in code, the target call site comprising a call to a routine;

establishing a breakpoint at an entry point to the selected target call site;

determining call context information identifying a location of the selected target call site in the code;

upon encountering the breakpoint during execution of the code, determining whether the routine is called from the selected target call site based on the call context information; and

if so, halting execution of the code and then performing a step into operation to step into the routine.

28. (Currently Amended) The computer readable storage medium of claim 27, wherein the call context information unambiguously identifies the code.

29. (Currently Amended) The computer readable storage medium of claim 27, wherein the code is object-oriented and the target call site is a method of the object-oriented code.

30. (Currently Amended) The computer readable storage medium of claim 27, wherein determining whether the routine is called from the selected target call site based on the ~~stored~~ call context information comprises comparing the ~~stored~~ call context information to selected content of a call stack.

31. (Currently Amended) The computer readable storage medium of claim 30, wherein the routine is determined to be called from the selected target call site if the stored call context information matches the selected content of the call stack.

32. (Currently Amended) The computer readable storage medium of claim 30, wherein the selected content of the call stack is a call to the routine and wherein the routine is determined to be called from the selected target call site if the stored call context information matches the selected content of the call stack.

33. (Currently Amended) A computer readable storage medium containing a program which, when executed, performs an operation for debugging object-oriented code, comprising:

receiving a selection of a target call site in the code, the target call site comprising a call to a routine method;
identifying a plurality of entry points for the method;
establishing a breakpoint at each of the plurality of entry points;
determining call context information uniquely identifying the selected target call site; and
for each of the breakpoints encountered during execution of the code:
determining whether the routine is called from the selected target call site based on the call context information; and
if so, halting execution of the code and then performing a step into operation to step into the routine.

34. (Currently Amended) The computer readable storage medium of claim 33, further comprising automatically removing each of the breakpoints upon determining that the routine is called from the selected target call site based on the call context information.

35. (Currently Amended) The computer readable storage medium of claim 33, wherein determining whether the routine is called from the selected target call site based on the ~~stored~~ call context information comprises comparing the ~~stored~~ call context information to selected content of a call stack.

36. (Currently Amended) The computer readable storage medium of claim 35, wherein the routine is determined to be called from the selected target call site if the stored call context information matches the selected content of the call stack.

37. (Currently Amended) The computer readable storage medium of claim 35, wherein the selected content of the call stack is a call to the routine and wherein the routine is determined to be called from the selected target call site if the stored call context information matches the selected content of the call stack.

38. (Currently Amended) A computer, comprising:

a memory;
code under debug resident in the memory, the code comprising at as least one target call site comprising a call to a routine, the target call site being selected by a user;
a breakpoint data structure resident in the memory and configured for storing at least context information indicating a location of the call within the code; and
a debugger program resident in the memory and configured to interrupt execution of the code under debug and then automatically perform a step into operation to step into the routine in response to encountering a breakpoint set on the routine if the routine is called from the target call site as determined with reference to the context information.

39. (Original) The system of claim 38, further comprising a caller data structure resident in the memory and configured for storing at least callers of routines in the code as encountered during an execution path; and wherein the debugger program is configured to determine whether the routine is called from the target call site by comparing the context information to a caller stored in the caller data structure.

40. (Original) The system of claim 38, wherein the code under debug is object-oriented.

41. (Original) The system of claim 38, wherein the routine of the target call site is a method and is defined for at least two objects; and wherein the debugger program is configured to place breakpoints at each entry point to the method.

Please add the following claim:

42. (New) The method of claim 14, wherein the code is object-oriented code.